2

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

AD-A257 935

...ORT DATE	3. REPORT TYPE AND DATES COVERED FINAL 1 Feb 90 - 31 May 92
4. TITLE AND SUBTITLE "AN INFORMATION THEORETIC APPROACH TO DISTRIBUTED INFERENCE AND LEARNING" (U)	5. FUNDING NUMBERS 61101E 7013/DARPA
6. AUTHOR(S) Dr. Padhraic Smyth JPL	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Jet Propulson Lab 4800 Oak Grove Drive Pasadena CA 91109	8. PERFORMING ORGANIZATION REPORT NUMBER AFOSR-TR 92 [illegible]
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR/NM Bldg 410 Bolling AFB DC 20332-6448	10. SPONSORING/MONITORING AGENCY REPORT NUMBER AFOSR-90-0199
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; Distribution unlimited	12b. DISTRIBUTION CODE UL

13. ABSTRACT (Maximum 200 words)

The work has focused on developing information-theoretic on probabilistic models for neural network computation. This theoretical basis is then used to develop novel hybrid network architectures, which combine 6 techniques from the fields of statistics and artificial intelligence with neural approaches. A number of significant results including identification of the general class of energy functions which lead to proper probability estimation, a new algorithm which builds hybrid rule-based network models from data, Markov random field theory and algorithms for constructing network models from large databases, new results on sparse Markov models, a new unsupervised/supervised learning algorithm with applications to computer vision problems, a novel recurrent network structure, and prototype VLSI hardware implementations of these ideas. A total of 30 technical papers have resulted from this grant.

14. SUBJECT TERMS			15. NUMBER OF PAGES 34
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std 239-18
298-102

An Information Theoretic Approach to Distributed Inference and Learning

Professor Rodney Goodman
Department of Electrical Engineering
California Institute of Technology
Pasadena, CA 91125

Dr. Padhraic Smyth
Communications Systems Research
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109

October 1, 1992

DARPA BAA#89-03: Neural Networks - Theory and Modelling.
Contact: Barbara Yoon

Final Technical Report on Contract AFOSR-90-0199
Covering the period Feb. 1st 1990 to May 31st 1992.
Contact: Steve Suddarth

The continuation of this work for the period 7/1/92 to 6/30/95 is now under ONR grant: N00014-92-J-1860
Contact: Thomas McKenna

Contents

Abstract

This report describes research work which was funded under grant number AFOSR-90-00199 during the period February 1st 1990 to May 31st 1992. Our work has focused on developing information-theoretic and probabilistic models for neural network computation. This theoretical basis is then used to develop novel *hybrid* network architectures, which combine techniques from the fields of statistics and artificial intelligence with neural approaches. The report describes a number of significant results including identification of the general class of energy functions which lead to proper probability estimation, a new algorithm which builds hybrid rule-based network models from data, Markov random field theory and algorithms for constructing network models from large databases, new results on sparse Markov models, a new hybrid unsupervised/supervised learning algorithm with applications to computer vision problems, a novel recurrent network structure, and prototype VLSI hardware implementations of these ideas. A Total of 30 technical papers have resulted from this grant.

1 Introduction

The general objective of this work is to develop information-theoretic and probabilistic models for neural network algorithms and architectures. The practical goal of the research is to use the theoretical models as a basis for developing *hybrid* algorithms which combine ideas from connectionism, artificial intelligence, and information theory, with particular emphasis on learning problems. A primary motivating factor in our work is the belief that *hybrid* architectures which are based on sound probabilistic principles, explicit knowledge representation, and parallel network models offer significant advantages over more conventional approaches to pattern recognition and machine learning.

The broad foundations of the work can be summarized as follows:

1. General information-theoretic criteria for evaluation and comparison of neural network models: in particular, better theoretical understanding of the roles of goodness-of-fit and complexity for classification problems. The work on rule-based models (Section 2) is a good example of this work.

2. The role of probability in improving our understanding of neural network learning and behaviour — an example of this approach is our general result on the class of objective functions which are necessary for a network to converge to accurate posterior probability estimates (Section 3). In addition, in Section 4 we discuss our work on linking local basis function networks with non-parametric kernel density estimation.

3. The use of *explicit knowledge representation* in neural network models — the hybrid rule-based network models of Section 2 and the fuzzy networks described in Section 7 describe in more detail the application of this idea.

4. Theory and algorithms for automatically determining network architectures from data, e.g., the Markov random field models described in Section 3

5. *Hybrid* solutions to difficult pattern recognition problems which integrate both conventional techniques and neural models to significantly improve performance. For example, Hidden Markov models plus neural networks for fault detection (Section 4), and the use of unsupervised learning, Gabor filters and rule-based networks for texture discrimination (Section 5).

6. Experimentation and validation of the theory and models on real-world problems: medical database applications in Section 1, a NASA fault diagnosis task in Section 4, segmentation of remotely-sensed images in Section 5, and VLSI hardware implementation of particular network models (Section 8).

2 Hybrid Probabilistic Rule-based Neural Networks

2.1 Introduction

Our work in this area has focused on developing networks which use an explicit rule-based representation to represent the knowledge in a network. In particular we use ***probabilistic rules*** as the main basis for our models, extending our earlier results in machine learning to the connectionist paradigm. Our rule-based models can be considered to be direct descendants of the early work of Uttley and also Minsky and Selfridge in the late 1950's and early 1960's when they tried to build "conditional probability computers" using either random conjuncts or an exponential number of units in the hidden layers. We have developed techniques which allow the automatic synthesis from data of a network model, where the number of hidden units or rules are traded off as a function of goodness-of-fit and complexity [1]. The primary feature of our model is the use of ***explicit*** knowledge representation whereby the user can easily comprehend the learned model and receive explanations of classification decisions — in comparison, models such as backpropagation-trained networks yield very little insight into the underlying decision process, limiting their acceptance in real-world applications.

We utilize an information theoretic approach to learning a model of the domain knowledge from examples. This model takes the form of a set of probabilistic conjunctive rules between discrete input evidence variables and output class variables. These rules are then mapped onto the weights and nodes of a feed forward neural network resulting in a ***directly*** specified architecture (Figure 1). The network acts a parallel Bayesian classifier which produces posterior probability estimates of the class variables.

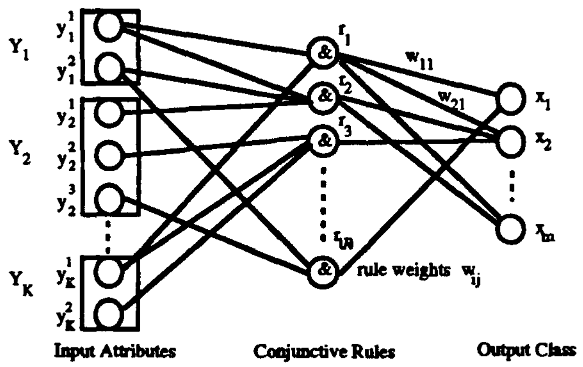

Figure 1: Architecture of the Rule-Based Classifier.

2.2 Models for Rule-based Neural Networks

We consider the problem of building a classifier that relates a set of K discrete feature variables (or ***attributes***) comprising the set $Y = \{Y_1, \ldots, Y_K\}$ to a discrete class variable X. Each attribute variable takes ***values*** in the alphabet $\{y_l^1, \ldots, y_l^{m_l}\}$, $1 \leq l \leq K$, where m_l is the cardinality of the lth attribute alphabet. The class variable X takes discrete values from the set $\{x_1, \ldots, x_m\}$, where m is the cardinality of the class. We also assume that we are given an initial labeled training set of N examples where each example is of the

J-Measure	Rule				Strength w_{ij}
0.297	IF	cell size uniformity 1	THEN	diagnosis benign	5.9
	AND	mitoses 1			
0.289	IF	bare nuclei 1	THEN	diagnosis benign	6.2
	AND	normal nucleoli 1			
0.271	IF	epithelial cell size 2	THEN	diagnosis benign	8.0
	AND	bare nuclei 1			
0.231	IF	bare nuclei 10	THEN	diagnosis malignant	-4.4
0.145	IF	clump thickness 10	THEN	diagnosis malignant	-5.7
0.111	IF	cell size uniformity 10	THEN	diagnosis malignant	-5.3
0.103	IF	normal nucleoli 10	THEN	diagnosis malignant	-5.2
0.085	IF	marginal adhesion 10	THEN	diagnosis malignant	-4.2
0.057	IF	cell size uniformity 5	THEN	diagnosis malignant	-4.5
0.056	IF	epithelial cell size 10	THEN	diagnosis malignant	-3.8
0.045	IF	bland chromatin 8	THEN	diagnosis malignant	-4.2

Table 1: Medical Database – Rules

form $\{Y_1 = y_1^a, \ldots, Y_K = y_K^z, X = x_i\}$. Let s_j be an arbitrary conjunction of left-hand side (LHS) or attribute terms, e.g., $s_j = y_1^a . y_4^b$. The supervised learning problem is to learn a classifier which when presented with future unseen attribute vectors (which may be either partial or complete) will estimate the posterior probability of each class.

In previous work we developed a measure of the utility or goodness of a probabilistic rule. In a Hebbian sense such a rule might be considered good if the occurrence of the LHS conjunction of variables is strongly correlated with the RHS. Alternatively, such a rule might be considered good if the transition probability p is near unity. For example, a rule with $p = 1$ indicates a deterministic rule in which the occurrence of s_j implies $X = x_i$ with certainty. Both aspects can be taken into account by considering that the goodness of such a rule can be measured by the average bits of information that the occurrence of the LHS s_j gives about the RHS $X = x_i$. This measure, called the J-measure, can be defined as:

$$J(X; s_j) = p(s_j)\left(p(x_i|s_j).\log\left(\frac{p(x_i|s_j)}{p(x_i)}\right) + (1 - p(x_i|s_j)).\log\left(\frac{(1 - p(x_i|s_j))}{(1 - p(x_i))}\right)\right) \qquad (1)$$

We have developed algorithms which use the J-measure to search a given training data set for informative rules. Classification of a new feature vector, given a set of rules, can be achieved by use of appropriate conditional independence assumptions as described in [1]. However, without constraining the search, there is a high likelihood that the rules will overfit the data. We have developed the following strategy: an initially large network of rules is grown in the first stage, which is then "pruned" back to a parsimonious network model during a second classifier construction stage. The quality of a given network is evaluated by using Rissanen's Minimum Description Length (MDL) criterion which quantitatively trades-off the goodness-of-fit (in terms of how many bits it takes to describe the data given the model) versus the complexity of the model. The final model is chosen by finding a local minimum of the MDL criterion in the search space.

2.3 Experimental Results

The algorithm is particularly good at finding very simple rule-networks which generalize well to new data — typically a network with only 5 or 10 rules is required for many well-known datasets. In [1] we demonstrate the application of this method to two widely-quoted datasets: the University of Wisconsin cancer database, and the protein secondary structure prediction problem. In each case, equivalent or better results than those previously reported in the literature were obtained, but using quite simple and *knowledge-explicit* models. Table 1 shows the actual rules in the network for the cancer problem while Figure 2 shows the final network architecture.

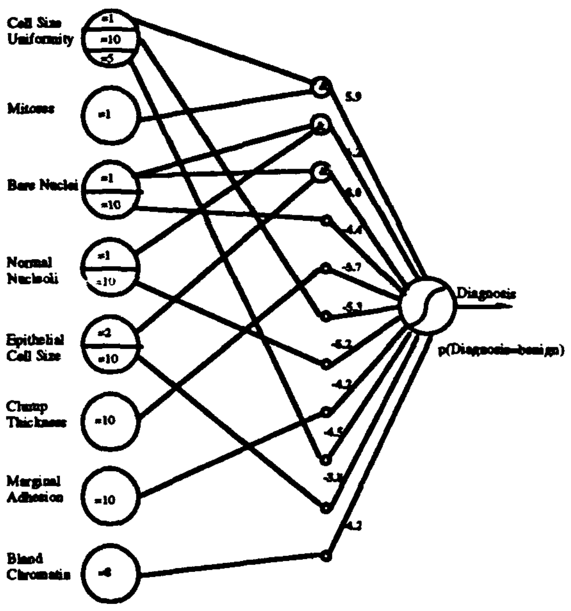

Figure 2: Medical Database – Network.

In many practical classification applications there is a strong preference on the part of the user to use a model which can in some sense justify its conclusion — the rule-based network can do this directly, not only providing an indication of which particular attributes were relevant to a given decision, but also providing the relative weight attached to each input. Furthermore, because the model is based on probabilistic rules it produces posterior probability estimates of the class labels directly, thus enabling a measure of confidence in the network's decision. As network models will be increasingly used as *components* in larger decision systems, the ability to produce coherent probabilistic outputs is critical.

2.4 Future Work and Applications

Because of the nature of the rule representation, the rule-based network model is most useful for dealing with discrete data. However, we are currently developing methods to handle continuous data. Other ongoing work involves the important problem of providing the ability to "pre-program" a rule-based network (this allows the advance specification of prior knowledge, and can considerably speed-up learning or equivalently reduce the amount of data required to learn a given task). There are many potential applications of the general model, in particular, problems where the user requires that the model provide a justification of how it arrived at a particular decision, for example, medical diagnosis systems and on-line monitoring of critical industrial, nuclear, or chemical processes. In Section 5 we discuss the application of the method to texture segmentation in image analysis, while in Section 7 the model provides the basis for learning fuzzy control strategies. We have also found that the method is quite useful in the early stages of any pattern recognition problem, since it provides useful feedback on which attributes are most relevant for discrimination as well as giving a general idea regarding the quality of the attributes relative to the given problem.

3 Probability Estimation in Neural Networks

3.1 Loss Functions for Posterior Probability Estimation

3.1.1 Introduction

We have proven a powerful result concerning the probability estimation properties of arbitrary objective functions used for training neural networks [2, 15, 17]. The result describes a unique class of functions which train the network so that its outputs provably correspond to probabilities — we describe such functions as being *P-admissible*. We show that the commonly used mean-squared error function is in a sense (in terms of description) the simplest member of the general class of P-admissible objective functions. In addition, the cross-entropy training metric, also widely used, and which can be derived from maximum likelihood arguments, is the only other easy-to-specify member of this general class. Thus the two best known objective functions are implicitly linked in a manner not previously known. The results also tells us that certain candidate objective functions which have been proposed in the literature (such as various L norms), are *not* in fact P-admissible probability estimators, and, as such, should not be used if the output activations are required to be interpretable as probabilities.

Backpropagation was originally derived in the context of minimizing a mean-squared error (MSE) objective function. More recently there has been interest in objective functions that provide accurate class probability estimates. The motivation of the work described here is to improve our understanding of systems which train iteratively to estimate probabilities. We derive necessary and sufficient conditions on the required form of an objective function to provide probability estimates. The results we present are discussed in the context of feed forward neural network models. The results, however, are dependent on the training scheme, not on the actual modeling scheme (such as a neural network, a decision tree, or a truth table).

3.1.2 Problem statement

We have a set of N samples where each sample consists of a vector of feature measurements and a class label (say there are K features and m possible class labels, $m \geq 2$). Define the class to be a discrete m-ary variable C, and let us refer to the K-dimensional feature variable as $\underline{x}$. From the training data we seek to infer a classifier, where a classifier takes as input an unlabeled feature vector and produces as output posterior probability estimates of the classes, i.e., an estimate of the conditional probability of each class given a particular feature value $\underline{x}$, $\hat{p}(c_i|\underline{x}), 1 \leq i \leq m$. We will find the following notation convenient: for each of the i training samples let $c_i(j)$ be the true class, i.e., the given class label.

We will refer to the estimated network parameters (weights and biases) collectively by $\underline{\theta}$. The most widely used error function is the Mean-Squared Error (MSE) function defined as

$$E_{MSE} = \sum_{i=1}^{N} \sum_{k=1}^{m} (t_i(k) - o_i(k))^2 \tag{2}$$

where $t_i(k)$ is the "target" value for node k and $o_i(k)$ is the network's output at node k ($o_i(k)$ is actually a function of the input features $\underline{x}(i)$ and the network parameters, but we ignore this dependence for notational convenience). Note that for labeled class data that $t_i(j) = 1, t_i(k) = 0, k \neq j, 1 \leq k \leq m$.

Two other objective functions have been proposed in the literature. The so-called cross-entropy (CE) measure is defined as

$$E_{CE} = \sum_{i=1}^{N}\left(\sum_{k=1}^{m} t_i(k)\log\frac{t_i(k)}{o_i(k)} + (1 - t_i(k))\log\frac{(1-t_i(k))}{1-o_i(k)}\right) \tag{3}$$

This definition is motivated by a desire to minimize the cross-entropy between the target distribution and the network estimate of the distribution for each class. In effect, it is a sum of binary cross-entropy measures for each node, rather than a true cross-entropy.

Consider now objective functions of the form

$$E = \sum_{i=1}^{N}\sum_{k=1}^{m} L(o_i(k), t_i(k)) \tag{4}$$

The function $L(y,t)$ is said to minimize to a probability when the its minimum is achieved when y is equal to $\bar{t}$, the average value of t taken over the training samples. The problem addressed in this work is to determine the necessary and sufficient conditions for minimization to a probability.

3.1.3 Results

It is found that minimization to a probability requires

$$L(y,t) = \int h'(y)\cdot\frac{y-t}{y}\cdot dy + C(t) \tag{5}$$

$$h'(y) > 0 \quad \text{for} \quad 0 < y < 1 \tag{6}$$

where $h(y)$ is any smooth function which satisfies (6). If the symmetry condition $L(y,t) = L(1-y, 1-t)$ is imposed then $h(y)$ must also satisfy (7):

$$\frac{1-y}{y} = \frac{h'(1-y)}{h'(y)} \tag{7}$$

Equation (5) ensures the existence of a local extremum at $y = \bar{t}$, while (6) forces this to be the unique minimum. It follows from (7) that at least one of the following cases must be true:

$h'(y)$ has a zero at $y = 0$ o $h'(y)$ has a pole at $y = 1$

The simplest functions satisfying the above restrictions are:

$$h'(y) = y \quad \Rightarrow \text{MSE} \tag{8}$$

$$h'(y) = \frac{1}{1-y} \quad \Rightarrow \text{CE} \tag{9}$$

We have extended these results to the practical case where the network does not possess sufficient representation power to model the input/output probability dependence exactly [2]. In this case we are interested in the ***approximation*** capabilities of the network. We have shown that all of the above results still hold in the approximation sense. The minimum achieved by the network will in fact be the best approximation to the desired probabilities, where "best" is measured in terms of the natural metric of the objective function itself, i.e., the squared error objective function minimises the mean-squared error distance between the network output and the desired minimum, and the cross-entropy function provides an optimal approximation as measured in terms of information distance between two probability distributions.

3.1.4 Applications

The results further support the use of the Cross Entropy and Mean Squared Error objective functions. The choice of a particular objective function depends on the individual estimation problem. Each of the MSE and CE loss functions for probability estimation have advantages and drawbacks in their own right. Using these equations one can choose CE, MSE, or a more complicated loss function appropriate to a specific task while insuring the property of minimization to a probability.

3.2 Markov Random Field Networks for Database Modeling

3.2.1 Introduction and Background

We have developed a novel algorithm for creating a neural network from a discrete database which produces accurate probability estimates as outputs [29]. The network implements a Gibbs probability distribution model of the training database. A database is viewed as a collection of independent samples from a probability distribution. Statistics are collected from the database and then a model is fitted to these statistics. The fitted model is a probability distribution. Samples from this distribution should, on average, have statistics identical to those collected from the original database. The problem can be separated into two parts: one is the choice of the statistics of interest. The second is the method of choosing a model which is consistent with these statistics. Under reasonable assumptions, the optimal solution to the second problem is the method of *Maximum Entropy*. For a broad class of statistics, the Maximum Entropy solution is a *Gibbs* probability distribution. In this section , the background and theoretical result of a transform from joint statistics to a Gibbs energy (or network weight) representation is presented. We then outline the experimental test results of an efficient algorithm implementing this transform without using gradient descent iteration.

Define a set T to be the set of attributes (or fields) in a database. For a particular entry (or record) of the database, define the associated set of attributes-values to be the configuration ω of the attributes. The attribute-values associated with a subset $b \subset T$ is called a subconfiguration ω_b. Using this set notation the Gibbs probability distribution may be defined:

$$p(\omega) = Z^{-1} \cdot e^{V(\omega)} \tag{10}$$

where

$$V(\omega) = \sum_{b \subset T} J_b(\omega) \tag{11}$$

The function V is called the ***energy***. The function J_b, called the ***potential function***, defines a real value for every subconfiguration of the set b. Z is the normalizing constant that makes the sum of probabilities of all configurations equal to unity.

Prior work in the neural network literature on producing Gibbs distribution models (such as the Boltzmann Machine) have primarily used second order potentials – Gibbs distributions with $J_b = 0$ if b is a set of more than two attributes, $|b| > 2$. By adding new attributes, second order potentials can be used to model increasingly complex distributions. The work presented in this paper, in contrast, uses higher order potentials to model complex prc˙ ıbility distributions.

The principal of Inclusion-Exclusion from set theory states that the following two equations are equivalent:

$$g(A) = \sum_{b \subset A} f(b) \tag{12}$$

$$f(A) = \sum_{b \subset A} (-1)^{|A-b|} g(b). \tag{13}$$

The method of inverting an equation from the form of (12) into one in the form of (13) is a special case of *Mobius Inversion*. Clifford-Hammersley used this relation to invert formula (11):

$$J_A(\omega) = \sum_{b \subset A} (-1)^{|A-b|} V_b(\omega) \tag{14}$$

Define the probability of a subconfiguration $p(\omega_b)$ to be the probability that the attributes in set b take on the values defined in the configuration ω. Using (10) to describe the probability distribution of subconfigurations, equation (14) can be written:

$$J_A(\omega) = \sum_{b \subset A} (-1)^{|A-b|} \, ln(\, p(\omega_b)\,) \tag{15}$$

Database	A	C	R	Train	Test	Trials	EM-Rate	Compare
House Voting	16	2	435	335	100	50	95.32%	95%
Letter Recognition	26	26	20000	16000	4000	3	88.93%	80%
Iris	4	3	150	149	1	100	97.1%	97.1%
Breast Cancer	9	2	369	200	169	100	95.7%	93.7%
Monks 3	6	2	432	122	432	1	97.5%	68-100%

A	=	Attribute count in the database, excluding the class attribute
C	=	Class count
R	=	Record count
Train	=	Number of records used to create the energy model for a single trial
Test	=	Number of records tested in a single trial
Trials	=	Number of independent train-test trials used
EM-Rate	=	Energy Model classification rate
Compare	=	Baseline classification result of other classification methods

Table 2: Summary of Energy Model Classification Results

3.2.2 Results

Equation (15) provides a technique for modeling distributions by energy functions rather than directly through the observable joint statistics of sets of attributes. If the model is truncated by setting high order potentials to zero, then the energy model becomes an estimate of the model obtained by collecting the joint statistics, rather than an exact equivalent. Our method of probability estimation is to first collect empirical frequencies of patterns (subconfigurations) from the database. An efficient hash table implementation of the algorithm (for a serial computer) collects the statistics without doing searches or multiple passes through the training dataset. Second, interpreting these frequencies as probabilities, we convert each pattern frequency to a potential using a normalized version of equation (14). Alternately, a distributed neural network implementation can use a modified hebbian rule to push the weights toward the desired potential value. We assume patterns with unknown or uncalculated frequencies have zero potential. Finally, we calculate the probability of any new pattern not in the training set using the network implementation of equations (10) and (11).

One way to validate the performance of a probability model is to test its performance as a classifier. The probability model is used as a classifier by calculating the probabilities of each unknown class value together with the known attribute values. The most probable combination is then chosen as the predicted class. Used as a classifier the Gibbs model tied or outperformed published results on a variety of databases. Table 1 outlines these results on three datasets taken from the UC Irvine archive.

3.2.3 Conclusion

A new method of extracting a Gibbs probability model from a database has been outlined. The approach uses the principal of Inclusion-Exclusion to invert a set of collected statistics into a set of potentials for a Gibbs energy model. A hash table implementation is used to efficiently process database records in order to collect the most important potentials, or weights, which can be stored in the available memory. Although the model is designed to give accurate probability estimates rather than simply class labels, the model in practice works well as a classifier on a variety of databases.

3.3 Theoretical Results on Minimum Description Length Models

3.3.1 Introduction

We have also investigated the trade-off between complexity and goodness-of-fit as it applies to neural network architecture selection. By looking at the posterior probability of a particular network model given a set of data, Bayesian and Minimum Description Length techniques reduce to two types of terms: a likelihood or goodness-of-fit term (the standard squared-error or cross-entropy objective function term) which measures how well the data fits the model, *plus* a second complexity penalty term which corresponds to a prior probability or description length of the model itself. The purpose of the second term is to quantitatively enforce Occam's razor — a preference for simpler theories or models for the data because they have a better chance of extrapolating well to new data.

3.3.2 Results

One of the problems with searching over multiple possible models is the computational complexity since there are so many possible models and parameters to choose from. We have developed a theory of ***admissible models*** in the context of MDL [4, 19]. This theory defines an admissible model to be one such that its complexity does not exceed that of the data itself. By using information-theoretic bounds, the boundaries of this class of models Ω can be calculated *a priori* with very little information about the data — even just knowing the number of sample points and the number of classes can lead to useful bounds. The practical consequence of this result is that the algorithm can thus restrict its search to within the boundaries of Ω. Not only does this reduce the search by orders of magnitude, it also reduces the probability that a poor local minimum will be found during the search by eliminating a large part of the search space at the start. We have applied the theory to some published results in the literature on Markov models and decision trees [4], and showed that the *chosen* model space Γ is often orders of magnitude larger than the necessary model space Ω. The bounds on Ω can be applied at any point of the learning algorithm leading to general branch-and-bound strategies for complexity-based learning algorithms. The theory has also been successfully demonstrated on the problem of finding the architecture of rule-based networks described earlier [19].

4 Probability Density Estimation and Neural Networks

4.1 Theoretical Analyses and Results

4.1.1 Introduction

We have investigated the links between statistical kernel density estimation and the recently proposed local basis function neural networks. This work was motivated by a practical problem: can one build a classifier which can not only discriminate between the classes on which it was trained, but also detect and reject data from "other" novel classes? This is a particularly important problem in applications such as medical diagnosis and fault diagnosis in complex systems, where the available training data may only consist of normal data with perhaps a few abnormal or fault classes, but certainly nothing like an exhaustive list of all possible classes.

4.1.2 Results

A variety of 'ad hoc' solutions to the novel class problem have been proposed in the past. However, a more principled approach is to formulate the problem in terms of Bayes' rule. From this vantage point, it is easy to show that, in general, *generative* models (which model the probability densities of the features conditioned on the classes) can identify novel classes, while *discriminative* models (which model the densities of the classes conditioned on the features) can *not* detect novel classes [9]. A good example of the latter is a feedforward network with a single hidden layer of sigmoidal units. As shown in the previous section, under the appropriate assumptions and with the appropriate loss function, the outputs of this network will tend to converge to the posterior class probabilities. However, if data from a novel class is presented at the input to the network, it is highly likely that it will respond with a strong activation (near 1) for one of the outputs and 0 for the others. Hence, the user would be completely unaware of the presence of novel data, a potentially dangerous situation in practice. The basic problem lies with the non-local nature of the representation being used in the model and the difficulties in extrapolation with such non-local models. In contrast, local basis functions offer the ability to detect changes in the input data, and, hence, in principle, offer a much broader basis for on-line learning algorithm.

There has been much recent work in applied statistics on non-parametric kernel density estimation and we have shown how some of this work can be profitably applied to developing local basis function models [9] (the equivalence between kernel models and local basis function networks has been well-known for some time). The difficulties in applying local basis function methods include the necessity for a well-defined metric in the feature space, choosing appropriate basis function widths, and sparsity of data in high dimensions. In our experimental work we have found that provided a reasonable metric exists (i.e., that the concept of *feature-scale* is well-defined), the problem of choosing appropriate widths can be solved relatively well and generative models (which are competitive with discriminative models in terms of discrimination ability) can readily be learned for most classification problems. In addition, basic results s not nearly so important as the width — this has useful consequences for practical implementation, since kernel functions can be chosen simply based on how easy they are to compute in software or to generate in hardware.

4.1.3 Future Work

The simple kernel-based generative models are not sufficiently powerful on their own to solve many difficult learning problems. *Hybrid* models which use both discriminative and generative components provide an interesting avenue for further exploration. In addition, given the well-known deterioration in kernel performance as dimensionality increases, techniques which can reduce the effective dimensionality of the original feature space are of direct interest. In conclusion, it may be conjectured that the trend in building useful classification systems will be towards models which have, at least as a component, a local memory-intensive representation — our work so far has shown that such a representation is a necessary component in a learning system in order to model interesting and realistic learning behaviour and that the local representation can provide useful discrimination capabilities compared with more standard alternatives.

4.2 Applications of Density Networks and Hidden Markov Models to Fault Diagnosis

4.2.1 Introduction

The original motivation for our work on generative local basis function models came from a real-world fault diagnosis problem. As part of NASA's Deep Space Communications network, the Jet Propulsion Laboratory operates 70-meter fully steerable ground antennas in California, Spain and Australia to provide 24-hour radio communication capability with various inter-planetary spacecraft. These large antennas are potential single-points of failure in the network and are over 20 years old. A fault detection system monitors the antenna pointing system in real-time and uses on-line sensor data such as motor currents, wind velocity, tachometer readings, and so forth, to classify the antenna state into normal and fault classes. There is significant motivation to be able to detect novel classes — from an operational point of view there is little tolerance for any detection system that might mistakenly classify a possibly catastrophic new fault as normal simply because it is on the same side of a decision boundary as the normal features.

4.2.2 Results and Significance

We have developed a hybrid signal processing/neural network/Hidden Markov model (HMM) for this problem which has proven very successful and has been described in detail in [5, 22, 24, 27] — this work has been co-funded by NASA since 1991. Figure 3 shows the posterior probability of the condition "normal" (as a function of time, and with ground truth being "normal") for the neural density model both with and without the HMM component: the non-HMM model is quite unreliable whereas the neural/HMM combination shows much better resistance to false alarms.

The key components which have made the difference between the model being just an interesting laboratory demonstration and a real-world application are:

1. The use of a coherent probabilistic model throughout, enabling the direct integration of the neural component with the other models.

2. Generative local basis function models to enable detection of non-normal, non-predicted behaviour.

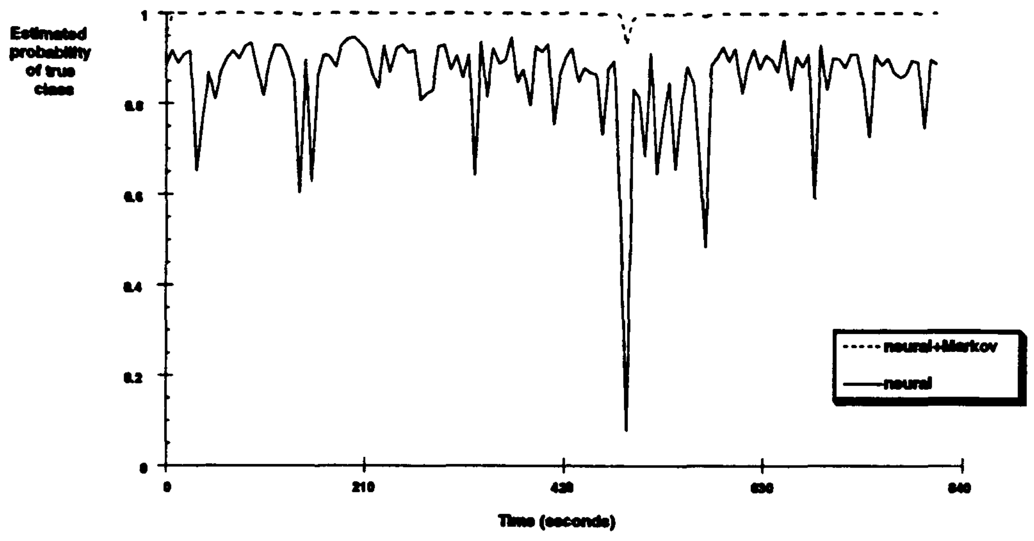

Figure 3: Effect of Combining Neural Network and HMM model

3. Hidden Markov models to reduce the false alarm rate without lowering the detection capability of the system — the HMM method has been adapted from other DARPA-sponsored work on neural/HMM models for speech recognition.

The general HMM monitoring model appears to be a significant practical advance over other techniques currently in practical use for on-line monitoring of dynamic systems. There is significant potential for the application of this general method to monitoring problems in medical health care, industrial plants, nuclear power industry, and similar problems where on-line health monitoring of critical systems is important.

5 Application of Rule-based Networks to Texture Modeling

5.1 General Goals

In this research we have developed a hybrid texture analysis system that incorporates the advantages of learning paradigms (including statistical machine learning, knowledge-based systems and neural networks) in the context of multi-resolution feature extraction techniques. The main goal of the system is to learn a minimal representation for a given library of textures, based on which one can successfully classify and segment new mosaic test images into homogeneous textured regions. Of particular interest is to apply the system to noisy images arising in real-world computer-vision problems.

5.2 Background

Visual texture is one of the most fundamental properties of a visible surface. As such it takes part in lower-level to higher-level tasks, from scene segmentation to object recognition. Texture-analysis methods can be utilized in a variety of application domains, such as remote sensing, automated inspection, medical image processing and advanced image-compression schemes. The different textures in an image are usually very apparent to a human observer but no good mathematical definition can capture the very diverse texture family. It is this lack of definition that makes automatic description or recognition of these patterns a very complex and as yet an unsolved problem.

Much effort has been expended to automatically segment and recognize different types of texture. Although researchers approach texture differently, most would agree that the texture family can be categorized into two main categories - *structured* and *unstructured*, more *stochastic* textures. Methods that can handle the more structured textures use structural models of texture which assume that textures are composed of texture primitives. The texture is produced by the placement of these primitives according to certain placement rules. One needs to be able to define a priori a good set of primitives and placement rules (a tree grammar is commonly used) in order to characterize the textured input. This approach can handle very regular patterns. Stochastic models, such as the Markov Random Field (MRF) models, are used as methods to handle unstructured or stochastic textures. Here the image is seen as an instance of a random process, defined via the model parameters. The model parameters need to be estimated in order to define adequately the perceived qualities of the texture. Synthetic textures can then be generated and compared to the original images. This model-based technique can capture certain textures very well but they fail with the more regular textures as well as inhomogeneous ones.

Although texture analysis has been a subject of intense study by many researchers, it is as yet an open challenge to achieve a high percentage classification rate on all the above textures within one framework. In this work we demonstrate the application of a learning system to the texture-analysis task [7, 11, 13, 16, 23, 30]. In this approach, the important characteristics of the input domain are learned from examples, rather than specified *a priori* via model-based schemes such as the structural or stochastic models mentioned above.

5.3 System Characteristics

The main features of the system are the following:
- A multi-resolution pyramidal approach is used as a computationally efficient feature-extraction scheme.

- The important characteristics of the input domain are learned from examples.
- Both unsupervised and supervised learning are utilized [16].
- An information theoretic technique enables the characterization of the most informative correlations between the input features and the texture class specification [13].
- These learned correlations are specified as discrimination rules which are available to the user and can enhance his or her knowledge of the input domain and the classification task at hand [23].
- The learned rules can be mapped onto a rule-based neural network and thus the classification scheme is parallelizable and suitable for implementation using special purpose neural-network hardware [1].
- The rule-based network provides probability estimates for the output classes rather than just a hard-decision label as its output. These probability estimates can be used for higher-level analysis, such as feedback for smoothing and the learning of an unknown class, the so called "pattern discovery" problem.

The system consists of three major stages. The first stage performs feature extraction and transforms the image space into an array of 15-dimensional feature vectors, each vector corresponding to a local window in the original image. We define an initial set of 15 filters and achieve a computationally efficient filtering scheme via the multi-resolution pyramidal approach. The learning mechanism shown next derives a minimal subset of the above filters which conveys sufficient information about the visual input for its differentiation and labeling. We reduce the feature space both in the unsupervised and supervised stages of analysis. In the unsupervised stage a vector quantization algorithm is used to cluster the continuous-valued input features. The supervised learning stage follows in which labeling of the input domain is achieved using a rule-based network. Here an information theoretic measure is utilized to find the most informative correlations between the attributes and the pattern class specification, while providing probability estimates for the output classes. Ultimately, a minimal representation for a library of patterns is learned in a training mode, following which the classification of new patterns is achieved. For a detailed description of the system see [7].

5.4 Simulation Results

The system was tested on both structured and unstructured natural textures, taken from the Brodatz library of natural textures. Experimental results on and Landsat images are also described.

An example of a five-class natural texture classification is shown in Figure 4. The mosaic is comprised of grass, raffia, herringbone weave, wood and wool (center square) textures. The input mosaic is presented (top left), followed by the labeled output map (top right) and the corresponding probability maps for a prelearned library of six textures (grass, raffia, wood, sand, herringbone weave and wool, left to right, top to bottom, respectively). The input poses a very difficult task which is challenging even to humans. Based on the probability maps (with white indicating probability closer to 1) the very satisfying result of the labeled output map is achieved. The five different regions have been identified and labeled correctly (in different shades of gray) with the boundaries between the regions very strongly evident. It is worth noting that the probabilistic approach enables the analysis of both structured textures (such as the wood, raffia and herringbone weave) and unstructured textures (such as the grass and wool).

Our most recent results pertain to the application of the system to the noisy environment of satellite and airborne imagery. An example of these images is presented in Fig. 5 (top to bottom, respectively). The Landsat input is an AVIRIS image of Pasadena, California. Here, the resolution is 20 meters per pixel. From this example image we can see that a major distinguishing characteristic is urban area vs. hilly surround. These are the two categories we intended to learn. The training set consists of a 128*128 image sample for each category. The test input is a 512*512 image which is very noisy, and because of its low resolution, very difficult to segment precisely into the two categories, even to our own perception. In the presented output (top right), the urban area is labeled in white, the hillside in gray, and unknown, undetermined areas are in darker gray. We see that a rough segmentation into the desired regions has been achieved. The probabilistic network managed to generalize in a noisy environment. The network's output allows for the identification of unknown, unspecified regions, in which more elaborate analysis can be pursued. The dark gray areas correspond to such regions; an example of which are the suburbs on the hill slopes (bottom right) which contain mixtures of the classes. An example of airborne image analysis is presented at the bottom of Fig. 5. The input image (left) is of much higher resolution and here it is evident that segmentation based on texture is of importance. Intensity-based segmentation, for example, would have difficulties with the shadows present around the bush areas. The classes learned are bush (output label dark gray), ground (output label gray) and a structured area, such as a field or the man-made structures (white). Here, the training was done on 128*128 image examples (one example per class). The input image is 800*800. In the result shown (right) we see that the three classes have been found and a rough segmentation into the three regions is achieved. Note in particular the detection of the three main structured areas in the image, including the man-made field, indicated in white. These results demonstrate the network's capability of generalization and robustness to noise in two complex real-world images.

5.5 Conclusions and Future Applications

We have demonstrated the capability of the system to achieve high-classification rates for both structured and unstructured textures. The main advantages of the learning approach are the ability to handle all types of textures within one framework, and to produce probability estimates for the output classes. A minimal feature set is learned and the classification rules are available for the user's information. The system can thus enhance the user's knowledge of the input domain via its own extracted rule knowledge base. Note that a segmentation of the image is achieved via the recognition process.

It is of interest to pursue the application of the system to natural-scenery analysis, such as the Landsat images. Other applications, such as autonomous navigation, can also benefit from this texture recognition and segmentation system. In such applications, texture can help in the characterization of the scenery (bush vs. gravel etc.), together with other modalities, such as stereo, color etc. The fusion of this system with other sources of information, within a probabilistic framework, is a challenging future goal. Very recent experiments indicate that the system can achieve scale and rotation-invariant recognition. We are currently pursuing this goal further.

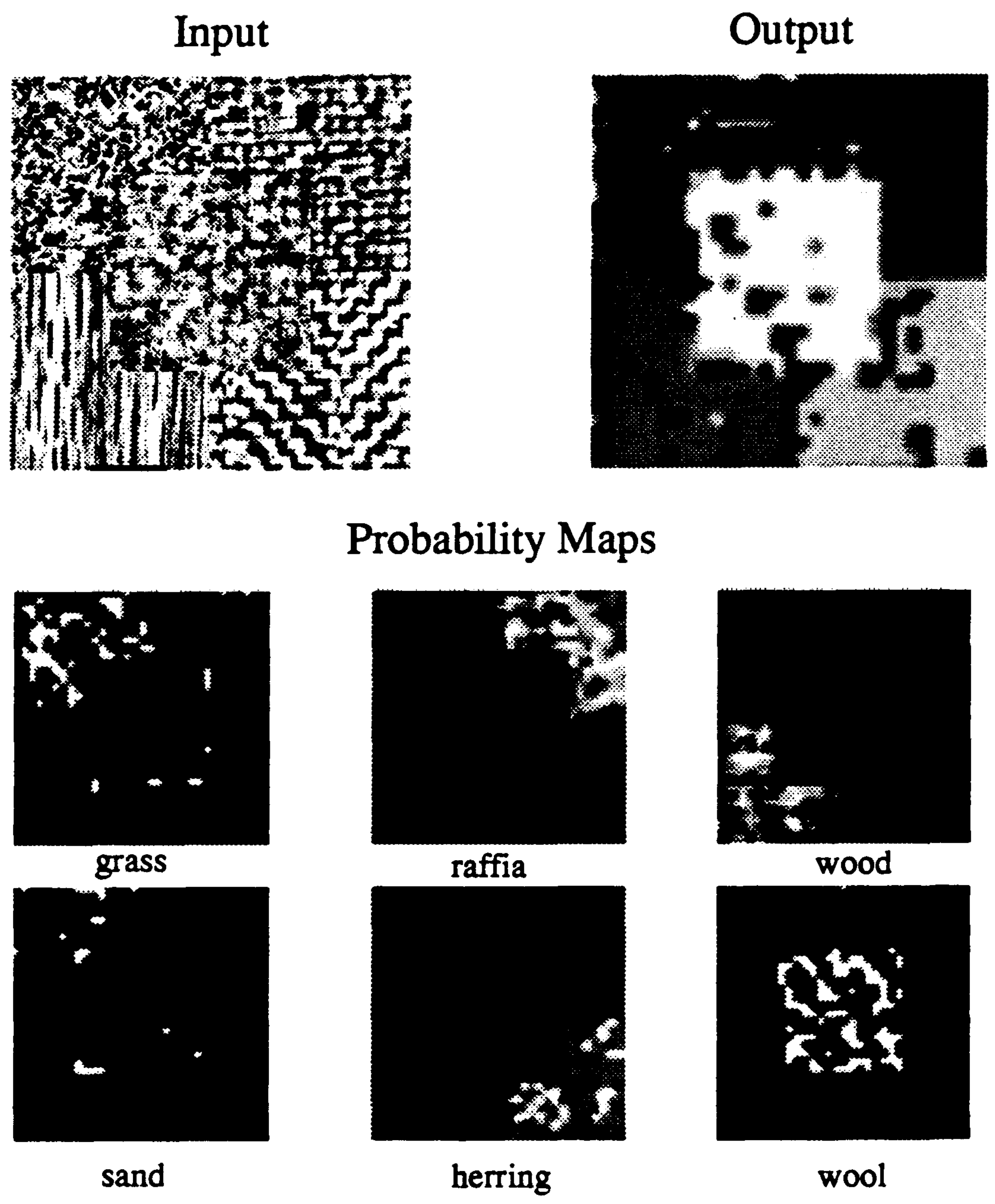

Figure 4: Five class natural texture classification. Input mosaic is presented (top left). followed by the labeled output map (top right) and probability maps (bottom). The probability maps correspond to a 6 - texture prelearned library. comprised of the (from top to bottom. left to right) grass. raffia. wood. sand. herringbone weave and wool textures. White areas indicate high probability. In the label map the different grey levels correspond to the 5 classes identified.

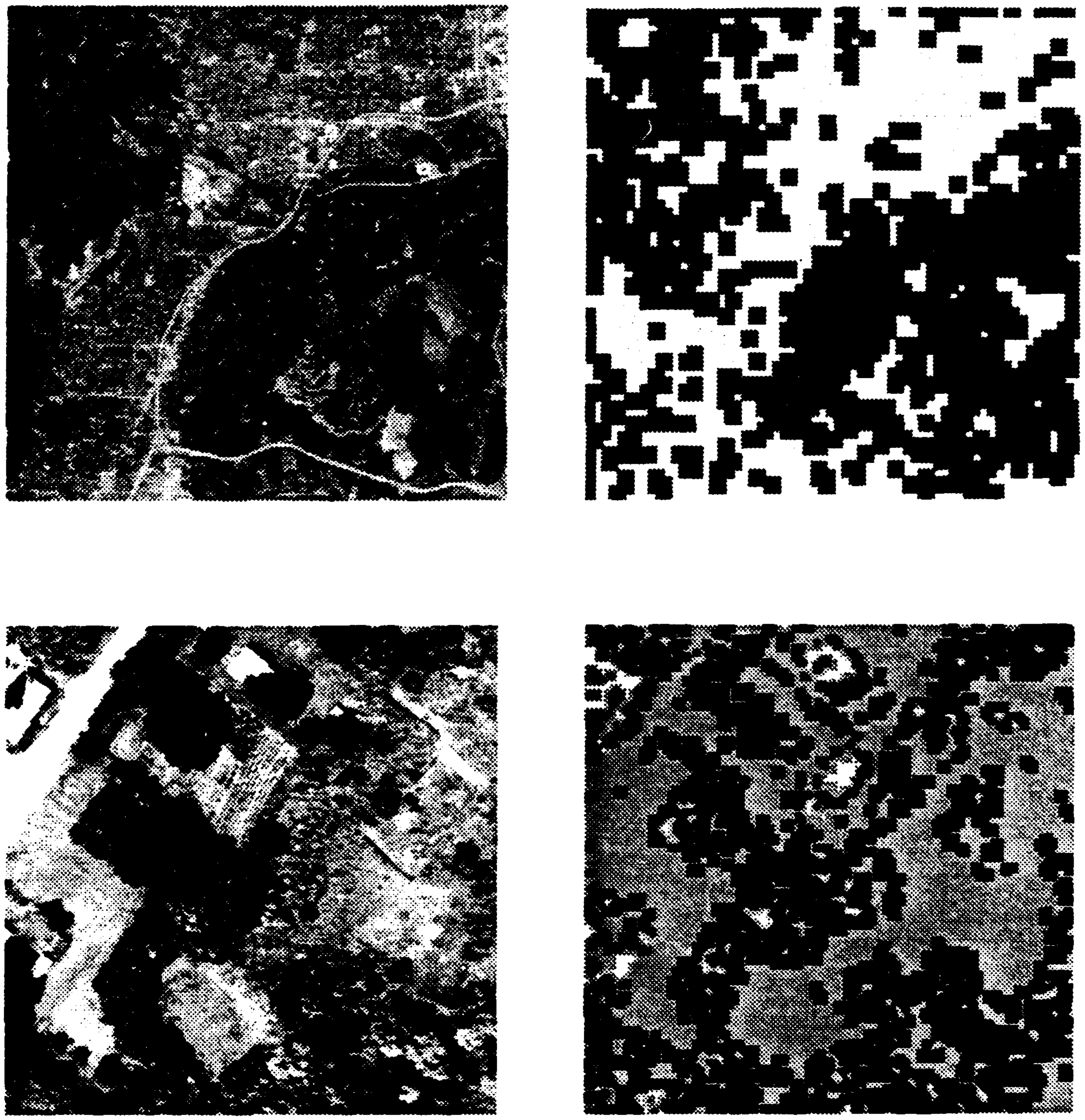

Figure 5: Landsat and Aerial image analysis results (top to bottom, respectively). The input test image is shown (left) followed by the system output classification map (right). In the Landsat output, white indicates urban regions, gray is a hilly area and dark gray reflects undetermined or different region types. In the Airborne output, dark gray indicates a bush area, light gray is a ground cover region and white indicates man-made structures. Both robustness to noise and generalization are demonstrated in these two challenging real-world problems.

6 Temporal Pattern Recognition Models

6.1 Sparse Markov Models

Motivated by our work both on rule-based models and model complexity, we conjectured that the standard approach of using a kth-order Markov model, where k is fixed (typically 1 or 2), in problems such as word recognition and image texture modelling, involves an unnecessarily large number of parameters. Based on the results in [4,19] one needs an inordinate amount of data in order to train such models and the number of parameters scales exponentially as a function of k. For problems where there are many states (such as word transition models in language modelling) the problem is particularly severe. Fully-parametrised *variable*-based models are over-complex compared with *event*-based models. In the context of Markov models this leads to a sparse representation for the model structure. High-order terms are represented by probabilistic rules up to arbitrary order, but the majority of the higher-order terms are not specified explicitly in the model. We applied this modelling technique to n-gram prediction for English text and were able to discover a very parsimonious set of rules [12]. In addition we showed how the model could be implemented in parallel on a network architecture both as a conventional feedforward network and as a pulse-firing network.

6.2 Self-Clustering Recurrent Networks

6.2.1 General Goals

Recurrent neural networks have recently been proposed and studied as a more powerful tool than straight feedforward neural networks for learning problems where information from the past is essential. However, there is little established theory for recurrent networks and their behavior can be very much unpredictable.

The focus of this work is to study in detail how a recurrent network could learn regular grammars. The purpose is to obtain a better understanding of recurrent neural networks, their behavior in learning, and their internal representations, which in turn may give us more insight into their capability for fulfilling other more complicated tasks, such as learning probabilistic grammars and time sequence predictions.

6.2.2 Problem Statement

A recurrent network is like a layered feedforword network but with feedback connections from higher layers to lower layers. In using recurrent networks to learn regular grammars (regular grammars are the simplest type of grammar in the Chomsky hierarchy and have a one-to-one correspondence to finite state machines), we define our problem as follows: The recurrent network is given a training set during learning. A training set consists of randomly chosen variable length strings with length uniformly distributed between 1 and L_{max}, where L_{max} is the maximum training string length. Each string is marked as "legal" or "illegal" according to the underlying grammar. The purpose is to train the network by looking at these labeled strings to behave like a finite state automaton that accepts the underlying grammar.

6.2.3 Background

Giles et al. have proposed a "2nd order" recurrent network structure to learn regular languages[1]. Our independent experiments have confirmed their results that 2nd order nets can learn various grammars well. However, a stability problem emerges with trained networks as longer and longer input strings are presented. The stability problem led us to look deeper into the internal representation of states in such a network and found that the network attempts to form clusters in activation space as its internal representation of states. However, these learned states become unstable as longer and longer strings are presented to the network.

To solve the stability problem we have developed a discretized combined network structure, as well as a pseudo gradient learning method, which can be shown to successfully learn stable state representations [6]. In the proposed network, instead of clusters, the states of the network are actually isolated points in hidden unit activation space.

6.2.4 Discretized Network Structures

During the learning procedure (which is a gradient descent method in weight space to minimize the mean squared error for each training string) we recorded the hidden unit activations at every time step of every training string in different training epochs. The following behavior was observed: during learning, the network attempts to form clusters in hidden unit space as its representation of states. Once formed, the clusters are stable for short strings, i.e., strings with length not much longer than L_{max}. However, as longer and longer strings are presented to the network, the percentage of strings correctly classified drops substantially. The well-separated clusters formed during training begin to merge together for longer and longer strings and eventually become indistinguishable. The problem can be considered as inherent to the structure of the network where it uses analog values to represent states, while the states in the underlying state machine are actually discrete.

We have developed a new method to force the network to learn stable states by introducing discretization into the feedback connections. A pseudo gradient algorithm is used to perform training. The algorithm uses the gradient of the soft sigmoid function (the pseudo gradient) as a replacement of the gradient of the hard-limiting function (which is zero almost everywhere), and thus provides a heuristic hint as to which direction and how close a step down or step up would be.

6.2.5 Summary of Experimental Results

For networks without discretization in the feedback path we found that they can successfully learn various grammars (2–10 states), with a small number of hidden units (4–5) and in less than 1000 epochs. But almost all the learned networks have problems with stability as described above. For networks with discretization added, and with the use of the pseudo gradient method, the learning performance is similar to that of the models without discretization, without the stability problems, i.e., classification rates are always 100% no matter how long the test strings are. In the discretized network, after learning, each point in the discretized activation space is automatically defined as a state. The transition rules are calculated as before, and an internal state machine in the network is thus constructed. In this manner, the network can be said to perform "self-clustering."

[1]C. L. Giles et al., 'Grammatical inference using second-order recurrent neural networks,' *Proceedings of the International Joint Conference on Neural Networks*, pp.273-281, Seattle, Washington, 1991.

7 Hybrid Fuzzy Rule-based Networks

7.1 General Goals

The unifying theme of this work has been to develop the application of fuzzy rule-based neural networks to function approximation from example data. This involves learning the rules to describe the data, and then building a neural network with the rules to predict the data. The rules are learned via information theory directly from the data. Rather than constructing a large network and altering the weights by a delta rule, we construct a network directly from the learned rules.

7.2 Problem Statement

The problem of function approximation is that of learning to predict a numerical output variable from a number of numerical input variables. The system is again given examples of the input and the corresponding output, but must construct a numerical model of the system in order to smoothly respond to any numerical input. It is our aim to apply this function approximator to learning neural-based control systems.

7.3 Background

Function approximation may be approached either by using a mathematical model of the system to be learned (the classical approach), or by using so-called "model-free" systems such as neural networks and fuzzy systems. Model-based systems can be extremely tedious for complex problems and require a fresh start on every new problem. Model-free systems, however, are applicable to a wide variety of approximation problems and require no special modifications. In neural networks, there are a number of well-known techniques for approximation, including radial basis functions and backpropagation. However, in neural network-based systems, once training is complete the function learned can be observed only through the input/output relationship. This drawback has led to widespread interest in the industry in fuzzy systems. A fuzzy system's learned function is expressed in terms of explicit rules, which can be observed and directly modified if unsatisfactory. While methods for fuzzy function approximation are not so well developed as those for neural networks, there are several distinct approaches, including clustering approaches to learning rules and back-propagation of membership functions and rules. The clustering approach requires the user to set up membership functions by hand. The backpropagation approach is very slow to converge and requires the user to set up the number of rules before training. In contrast, our method requires the user only to state the number of membership functions – the membership functions themselves and the rules are learned without human intervention. In addition, the entire learning process is based upon an algorithm which must terminate in a rather small number of steps, not depending on mathematical error convergence.

7.4 Summary of results

We have shown that our fuzzy function approximator is capable of learning a control function from example data without human intervention, and then using that function to reproduce control performance. We have done this for a complex problem in simulation (the truck backer-upper), the well-known pole-balancing problem (simulated), and a simpler problem in real-time (cruise controller). In both cases, a network is built directly from the rules and membership functions. These results are described in detail in [8].

As an example, we summarize our results for the truck backer-upper problem. Jenkins and Yuhas [2] have developed by hand a very efficient neural network for solving the problem of backing up a truck and trailer to a loading dock. Its trajectory is nearly optimal by most criteria· RMS docking error, smoothness of path, energy cost in turning the truck, and so forth. We have chosen this system as a function to approximate because it is highly nonlinear and its features are non-parallel to the axes; this makes a more challenging approximation problem for our system.

The function approximator system was trained on 225 example runs of the Yuhas controller, with initial positions distributed symmetrically about the field in which the truck operates. While the learned fuzzy system with 5 truck angle membership functions actually performs better in RMS docking error than the original Yuhas network, its path is sometimes not as smooth. The fuzzy truck backer-upper has "modes" of operation: the truck will first turn around, then back up in a straight line at a diagonal angle, then change direction sharply and back towards the loading dock. This is directly related to the piecewise approximation to the original function.

7.5 Applications

Given a function approximator, it is possible to learn the *inverse* of plant - that is, given the last state and the current state, to predict what the input was. Once this is accomplished, the current state and the desired state can be fed into the system to output the correct control signal. This method, known as direct inverse control, will be an important application of our system to unsupervised control.

Adaptation of a learned control system to changes in plant parameters is essential in a real control system. We are researching how to modify the rule weights and membership functions in response to a performance measure.

We currently have in the lab a helicopter which we intend to do control experiments on. Controlling even a hover of a helicopter requires multi-input linked multi-output control. Since we have at this time only dealt with single-output systems, this presents a new challenge for future work.

[2] 'A simplified neural-network solution through problem decomposition' preprint, 1992.

8 VLSI Implementation of Probabilistic Neural Networks

We are actively pursuing hardware implementations in VLSI of several of the ideas described above. Given that we have significant VLSI expertise and facilities within the group (Professor Goodman and his graduate students have successfully been fabricating signal processing and communications chips for several years), we can readily test whether or not a particular algorithm or technique is suited for direct implementation as a special-purpose VLSI chip.

8.1 Pseudo K-Means Clustering In Analog VLSI

8.1.1 Introduction

We have developed an analog VLSI system which adaptively clusters data using a modified K-means algorithm to act as either a pre-processor or post-processor for a neural network-based system. The clustering algorithm implemented is a variant of the traditional K-means algorithm optimized for realization in silicon. The algorithm offers the advantage of a fast unsupervised method for reducing data complexity. We have successfully verified this approach with both computer simulations and with a chip that has been fabricated and tested.

8.1.2 Motivation

The K-means algorithm is a traditional pattern classification algorithm that does data analysis based on a simple centroid-based method of clustering data. Our goal with this architecture is to augment learning systems by adding information about the clustering of data. Clustering can be applied to the outputs of a heterogeneous neural network to allow better discrimination of the classification information from the previous layers. The data storage reduction achieved by the K-means clustering algorithm can be large for well-behaved data because only the cluster centers and the number of points in each one need to be stored to entirely specify the clustering.

8.1.3 Modifications to the Standard K-means Algorithm

The K-means algorithm starts out by defining K clusters, each defined by its centroid: $C_j \quad 1 \leq j \leq K$. In an iteration of the algorithm, the data is individually grouped into one of the clusters based on the closeness of the data to a particular centroid in some metric (typically euclidean distance or the city-block distance).

In order to feasibly implement the algorithm in analog VLSI, we have modified the algorithm somewhat. The first change is that in order to terminate the learning phase of the algorithm, we calculate how far the cluster centers have moved since the last iteration. If they have all moved below some user–settable threshold, we are finished with the cluster–movement part of the algorithm. The second change to the algorithm is that new clusters can be started during training whereas in the original algorithm, the number of clusters is specified *a-priori*.

The third change is probably the most important from the view of hardware implementation. Since we cannot easily afford to store all the data points associated with each cluster in the generically available VLSI technology (e.g. 2μm double metal CMOS through MOSIS). But we know that as the number of points in the cluster increase, the effect of any one point will proportionally decrease. Therefore, we have an architecture whereby we keep track of the number of points that have been assigned to the cluster in question and use this value to weight the effect of new points assigned to the cluster. This is easy to do in analog VLSI, and the behavior will mimic that of the original algorithm for new points added to the cluster.

The following equation describes what we would like to achieve with our modification:

$$C_j^p = \sum_{i=1}^{i=P} x_i^p \, b \, e^{-aN_j} \quad \textit{where } a, b \textit{ are constants} \tag{16}$$

We achieve this functional relationship by using the exponential current from voltage relationship in a weak-inversion biased MOS transistor. If we can linearly change the gate voltage (V_g) of a MOS transistor, then according to the following equation, we get an exponential change in the drain current I_d:

$$I_d = I_o \frac{W}{L} e^{\frac{\kappa V_g - V_s}{U_t}} \qquad U_t \approx 26 \; mV \quad \kappa \approx 0.7 \tag{17}$$

Hence, by storing the number of points assigned in a linear fashion and using this value to bias a MOS transistor's gate in weak inversion, we can use the current through the transistor to perform the exponential weighting.

8.1.4 Test Results

We have designed, fabricated, and tested a proof of concept VLSI clustering chip. The chip clusters data into two clusters and operates on two dimensional data. Because of the simple design the testing was constrained to ensure that the capacitor voltage did not change significantly during operation as to permit other parts of the chip to compute correctly. Test results have validated the ability of the distance (bump) circuit to produce an output which allows good discrimination of different inputs. We have found that the constant part of the input varies over approximately 3.2 volts on common mode input for different values of the bias voltage (which determines the scaling factor for each dimension). The test results have also shown that the chip is capable of clustering as designed. even with the addition of small amount of noise ($<$ 150mv).

8.1.5 Conclusion

We have developed and implemented a VLSI system that implements a K-means type clustering algorithm. From both simulations and the actual chip testing, we are confident that a more powerful clustering chip is quite feasible and are currently designing such a chip.

8.2 A Hebbian Learning Chip

8.2.1 Introduction and Design

The goal of this work is to recreate realistic dynamics of Hebbian learning in VLSI using switched capacitor circuits for delayed dendritic connections and synapses. Neural activity in this architecture is frequency-coded. Inhibition is allowed, but there is no *negative* activity. In the architecture all outputs initially receive uncorrelated noise. All weights are initialized to zero; they are neither inhibitory nor excitatory. As input neurons receive training vectors, correlations resulting from random circuit mismatches and input-output synchronization due to noise lead to the updating of synapses. Synapses connecting correlated input-output pairs gain charge, moving towards positive excitatory weights. Synapses connecting input-output pairs with negative correlation lose charge, moving towards negative inhibitory weights. The uncorrelated noise that the output neurons initially receive is thus essential in pairing an input cluster with an output neuron. While output neurons inhibit one another, input vectors are expected to be clustered into separate sets, the number of which is determined by the number of mutually inhibiting sets of output neurons.

8.2.2 A Description of Network Dynamics

As the network dynamics evolve, an input neuron must control its output in proportion to both its oscillation frequency and the the strength of the connecting synapse. All output neurons ought to receive noise. This implies that there must exist a background random activity associated with uncommitted output neurons. Such noise could be generated by averaging linear-feedback shift register outputs. These activities between output neurons must be uncorrelated. As synapses strengthen and settle into their final values, this noise will become quite negligible. Consequently, following the **learning** process, any output neuron activity is almost exclusively controlled by its inputs and the strength of the connecting synapses. For most unsupervised learning schemes and self-organizing networks in particular, the process of synaptic update is guided in part by the interaction between the neurons of the output layer. In our implementation, the interaction in the form of lateral inhibition is controllable by the use of the dendritic delay element between neural oscillations and inhibitory input. This implies that higher frequency oscillations cause more effective inhibition. A possible way of implementing neighbor-excitation would be to supply correlated noise to neighboring neurons as well as electrically coupling them.

8.2.3 Results and Future Applications

A chip incorporating the building blocks of the architecture (specifically the input neuron, dendrite, synapse and output neuron) has been designed, fabricated and successfully tested. We envision that this architecture can be used in applications where the whole system needs to reconfigure itself in real time based on incoming sensory data.

8.3 A Digital Neural Network Using Random Pulse Trains

8.3.1 Introduction and Network Architecture

The motivation is to implement a pulse-train feed-forward neural network using very little chip area and only digital VLSI technology. We have focused on the multiplication component of a general pulse network architecture. The circuit consists of a digital delay line to simulate the phase shifted weight and input activity level representations. The input to the delay line is provided from a pulse width modulator that is driven by an analog input value. This particular method of mixing analog and digital techniques was chosen to test this multiplexing scheme over a continuous range of values. The random bit streams are generated by a shift register sequence, based on irreducible polynomials. This is done to obtain the pseudorandom bit sequence with the longest possible period. The bit streams are used to select between the four phase shifted versions of the sculpturing weight and the input. These signals are AND'ed with each other, and the result gets time averaged using a simple RC circuit.

The AND multiplication component has been tested both with one multiplicand fixed and both multiplicands equal. In both of these cases the chip has performed in a robust manner over frequency ratios ranging from 0.1 to 10. This has been demonstrated by time averaging the pulse trains through an RC filter with a fixed time constant (0.1 sec.). The frequency of activity levels are adjusted by changing the ratio of frequencies between the triangular wave generating the pulse-width-modulated representation and the clock generating the uncorrelated bit sequences which multiplex between them In the proposed digital architecture, this would correspond to the frequency ratio between the clock recycling the partial weight representations and the clock generating the uncorrelated bit sequences.

8.3.2 Results and Future Applications

The above design has been fabricated and successfully tested - details are reported on in [26]. The architecture described above could be used as a stackable fully-interconnected feed-forward neural network. An additional advantage is that the synaptic accuracy is controllable which has potential advantages for implementing stochastic learning algorithms. We are currently investigating the mapping of our rule-based learning algorithms and networks (described earlier) to this hardware architecture for fast parallel implementation.

9 List of Papers written during Grant Period

9.1 Journal Papers

1. R. Goodman, P. Smyth, J. W. Miller and C. Higgins, 'Rule-based neural networks for classification and probability estimation,' *Neural Computation*, accepted for publication, February 1992.

2. J. W. Miller, R. Goodman, and P. Smyth, 'On loss functions which minimise to conditional expected values and posterior probabilities,' *IEEE Transactions on Information Theory*, accepted for publication, April 1992.

3. R. Goodman and P. Smyth, 'Automated induction of rule-based neural networks from databases,' *International Journal of Intelligent Systems in Accounting, Finance and Management*, accepted for publication, April 1992.

4. P. Smyth, 'Admissible stochastic complexity models for classification problems,' *Statistics and Computing*, **2**, 97–104, 1992.

5. P. Smyth, 'Hidden Markov models for fault detection in dynamic systems,' submitted to *Pattern Recognition*, May 1992.

6. Z. Zheng, R. Goodman, and P. Smyth, 'Learning finite-state machines with self-clustering recurrent networks,' submitted to *Neural Computation*, May 1992.

7. H. Greenspan, R. Goodman, R. Chellappa and C. Anderson, 'Learning texture discrimination rules in a multiresolution system," submitted to the special issue on 'Learning in Computer Vision" of the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, July 1992.

8. C. M. Higgins and R. Goodman, 'Fuzzy rule-based networks for control,' submitted to *IEEE Transactions on Fuzzy Systems*, August 1992.

9.2 Book Chapters

9. P. Smyth, 'Probability density estimation and local basis function neural networks,' in *Computational Learning Theory and Natural Learning Systems*, T. Petcshe, M. Kearns, S. Hanson, R. Rivest (eds), Cambridge, MA: MIT Press, to appear, 1992.

10. P. Smyth, 'Admissible stochastic complexity models for classification problems,' in *Artificial Intelligence Frontiers in Statistics: AI and Statistics 3*, D. Hand (ed.), Chapman & Hall: London, to appear, 1993.

11. H. Greenspan and R. Goodman, "Neural Network Texture Recognition," submitted as a chapter in *Neural Network Technology Applications*, 1992.

9.3 Conference Papers

12. R. Goodman and P. Smyth, 'Sparse Markov models,' presented at *Neural Networks for Computing*, Snowbird, April 1991.

13. H. Greenspan, R. Goodman, 'Texture classification using information theory,' presented at the IEEE International Symposium on Informa tion Theory, Budapest, June 1991.

14. C. M. Higgins and R. Goodman, 'Incremental rule-based learning,' presented at the IEEE International Symposium on Information Theory, Budapest, June 1991.

15. R. Goodman, J. W. Miller, and P. Smyth, 'Objective functions for neural network classifier design,' presented at the IEEE International Symposium on Information Theory, Budapest, June 1991.

16. H. Greenspan, R. Goodman and R. Chellappa, 'Texture analysis via unsupervised and supervised learning,' in *Proceedings of the 1991 International Joint Conference on Neural Networks*, Seattle, July 1991.

17. J. W. Miller, R. Goodman, and P. Smyth, 'Objective functions for probability estimation,' in *Proceedings of the 1991 International Joint Conference on Neural Networks*, Seattle, July 1991.

18. C. M. Higgins and R. Goodman, 'Incremental learning using rule-based neural networks,' *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, 875-880, July 1991.

19. P. Smyth, 'On admissible stochastic complexity models for neural network classifiers,' poster presentation at the IEEE Neural Information Processing Systems Conference, Denver, CO, December 1990: also in *Advances in Neural Information Processing Systems 3*, R. Lippmann, J. Moody, D. Touretzky (eds.), Morgan Kaufmann, July 1991.

20. C. M. Higgins and R. Goodman, 'Fuzzy control using rule-based neural networks,' presented at *The Second Government Neural Network Applications Workshop*, Huntsville, Alabama, September 10-12, 1991..

21. R. Goodman and P. Smyth, 'Automated induction of rule-based neural networks from databases,' invited presentation, 4th International Symposium on Expert Systems in Business, Finance, and Accounting, Pasadena, CA, October 1991.

22. P. Smyth and J. Mellstrom, 'Fault diagnosis of antenna pointing systems using hybrid neural networks and signal processing techniques,' presented at the IEEE Neural Information Processing Systems Conference, Denver, CO, December 1991: also in *Advances in Neural Information Processing Systems 4*, J. E. Moody, S. J. Hanson, and R. P. Lippman (eds.), Morgan Kaufmann Publishers: Los Altos, CA, 1992.

23. H. Greenspan and R. Goodman, 'A combined neural network and rule-based framework for probabilistic pattern recognition and discovery,' poster presentation at the IEEE Neural Information Processing Systems Conference, Denver, CO, December 1991: in *Advances in Neural Information Processing Systems 4*, J. E. Moody, S. J. Hanson, and R. P. Lippman (eds.), Morgan Kaufmann Publishers: Los Altos, CA, 444-452, 1992.

24. P. Smyth, 'Hidden Markov models for decision-making over time,' invited presentation, Second Annual Workshop on Normative Systems, USC, March 1992.

25. C. M. Higgins and R. Goodman, 'Learning fuzzy rule-based neural networks for function approximation,' in *Proceedings of the International Joint Conference on Neural Networks*, Baltimore, pp 251–256, June 1992.

26. G. Ertzen and R. Goodman, 'A digital neural network using random pulse trains,' in *Proceedings of the International Joint Conference on Neural Networks*, Baltimore, June 1992.

27. P. Smyth and J. Mellstrom, 'Detecting novel classes with applications to fault diagnosis,' in *Proceedings of the Ninth International Conference on Machine Learning*, Morgan Kaufmann Publishers: Los Altos, CA, 1992, pp.416–425.

28. C. M. Higgins and R. Goodman, 'Learning fuzzy rule-based neural networks for control,' accepted for 1992 Neural Information Processing Conference, Denver, December 1992.

29. J. W. Miller and R. Goodman, 'Probability estimation from a database using a Gibbs energy model,' accepted for 1992 Neural Information Processing Conference, Denver, December 1992.

30. H. Greenspan and R. Goodman, 'Landsat image analysis via a texture classification neural network,' accepted for 1992 Neural Information Processing Conference, Denver, December 1992.

10 Personnel associated with work

- Professor Rodney Goodman, Associate Professor, Electrical Engineering Department, Caltech.
- Dr. Padhraic Smyth, Technical Group Leader, Communication Systems Research, Jet Propulsion Laboratory.
- Dr. John Miller, Graduate student until June 1992 , Electrical Engineering Department, Caltech: now at Microsoft Research Labs, Seattle, WA.
- Hayit Greenspan, Graduate student, Electrical Engineering Department, Caltech.
- Chuck Higgins, Graduate student, Electrical Engineering Department, Caltech.
- Bhusan Gupta, Graduate student, Electrical Engineering Department, Caltech.
- Zeng Zheng, Graduate student, Electrical Engineering Department, Caltech.
- Gamze Ertzen, Graduate student, Electrical Engineering Department, Caltech.